

インサイトの商品・サービス +AI・データサイエンス活動

2024/05/25

JSCEAM2024

Corporate Seminar @Room F

Insight, Inc.

Akio Miyoshi

目次

1. LLMの計算力学教育への適用トライアル(7)
2. 生成AI チャット経験の紹介(プログラミングや計算力学)(8)
3. Github copilotチャット経験の紹介(6)
4. Meshman_ParticleViewer_HPCの最新版紹介(3)
5. 計算力学2級スマホアプリの更新版紹介(2)
6. Meshman_ParticlePackingとOSSのDEMアプリ(2)
7. データサイエンス勉強会の更新版紹介(2)

1. LLMの計算力学教育への 適用トライアル(1)

- LLM(Large Language Model)
- Chat GPTが有名だが、他にも有る。
- 弊社で試した物:
 - ELYZA-japanese-Llama-2-7b
 - Phi-3
- 試した工夫
 - fine tuning
 - RAG(Retrieval Augmented Generation)
 - role(役割)の指定

1. LLMの計算力学教育への 適用トライアル(2)

- LLM選定のポイント
 - ライセンス形態
 - 日本語対応
 - サイズ

1. LLMの計算力学教育への適用トライアル(3)

- LLM利用の切っ掛け
 - 計算力学資格取得支援テキストのスマホ化
 - 知識データベースが、ワープロからデータ処理しやすい形となった
 - 合格対策テキストの更なる高度化に使えないか？

1. LLMの計算力学教育への適用トライアル(4)

プロンプト：“「ニュートン・コーツの積分法」について教えて”
Elyza + RAG

テキスト該当箇所

(b)ニュートン・コーツの積分は、表面上での数値評価が可能のため、板の曲げ応力評価において、良く使われる。

6.2.2 ニュートン・コーツの積分法

梁要素や板要素では、曲げ変形を取り扱うが、曲げ応力は表面に最大・最小値が現れるので、表面(要素の端面)に積分点を有するニュートン・コーツの積分法が用いられる事が多い。この積分は、ガウスの積分よりも効率が悪く、 n 点の積分点で $(n - 1)$ 次迄の多項式が完全に積分出来る。



与られた文章を元に、質問に回答します。

ニュートン・コーツの積分は表面上で数値評価が可能であり、板の曲げ応力評価に適していると記載されています。よって、質問に対する回答は「板の曲げ応力評価」です。

Phi-3 + RAG



ニュートン・コーツの積分法は、表面上での数値評価が可能のため、特に板や六面体の曲げ応力評価に適しています。この方法は、表面に積分点を配置し、曲げ応力を数値で計算することが一般的です。しかし、積分次数を増やすことで、結果の精度が向上するという点はありません。よって、ニュートン・コーツの積分法は、表面に積分点を配置した数値積分を用いることに特化しています。

1. LLMの計算力学教育への適用トライアル(5)

問題作成: ChatGPT 4o

固体力学に関するテキストから抜粋した一節を、下の「テキストからの抜粋」に示します。その内容に関して、何らかの練習問題を、以下の「フォーマット」に従って作成してください。ただし、テキストの内容はマークダウン言語で書かれています。また、出力する問題の内容はマークダウン言語で作成し、レンダリングされないように文字列として出力してください。

```
# テキストからの抜粋
## 3.4 曲げ応力
梁の曲げ応力を考える際に、一つの仮定を設ける。即ち「梁の軸線に垂直な断面(=横断面)は変形後も平面を保ちかつ軸線に垂直である」という仮定である。
----- 途中略 -----
式(3.4-2)を式(3.4-3)に代入して整理すると、結局 $\sigma$ と $M$ との関係は、 $\sigma = \frac{M}{I_z} y$  (式(3.4-4))
但し、 $I_z = \int_A y^2 dA$ であり、断面二次モーメントと呼ぶ。
```

```
# 問題のフォーマット
```text
問題:
{問題の内容}
```

```
選択肢:
A. {選択肢1}

B. {選択肢2}

C. {選択肢3}

D. {選択肢4}

```
```

問題:

式(3.4-2)における応力 $\sigma(y)$ を求めるために必要な情報はどれか。

選択肢:

- A. 曲げモーメント M と曲率半径 R
- B. 応力 σ と断面の幅 b
- C. 弾性係数 E と曲率半径 R
- D. 曲げモーメント M と断面二次モーメント I_z

問題:

曲率半径 R の役割として正しいものはどれか。

選択肢:

- A. 曲げ応力の計算に直接関与する。
- B. 曲げ変形の均一性を評価するための指標である。
- C. 曲げモーメントを決定するための値である。
- D. 中立面の位置を決定するための値である。

1. LLMの計算力学教育への適用トライアル(6)

問題作成: Phi-3 + RAG

「曲げ応力」に関して、何らかの練習問題を、以下の「フォーマット」に従って作成してください。ただし、出力する問題の内容はマークダウン言語で作成し、文字列として出力してください。

フォーマット

```
```text
```

```
問題:
```

```
{問題の内容}
```

```
選択肢:
```

```
A. {選択肢1}

```

```
B. {選択肢2}

```

```
C. {選択肢3}

```

```
D. {選択肢4}

```

```
```
```

問題:

梁の片持ち梁に対する最大曲げ応力を求める際、断面係数 Z を計算する際に、断面の形状が等高の円と等分の円のどちらかである場合、どちらの断面係数を使用するべきか？

選択肢:

A. 等高の円の場合、 $Z = I/(h/2)$

B. 等分の円の場合、 $Z = I/(h/2)$

C. 等高の円の場合、 $Z = I/(h/2)$

D. 等分の円の場合、 $Z = I/(h/2)$

問題:

梁の片持ち梁に対する最大曲げ応力を求める際、断面係数 Z を計算する際に、断面の対称性を考慮に入れた上で、断面の直径が300mm、断面の高さが150mm、そして梁の剛性比 EI が20000N/mm²である場合は、最大曲げ応力はどれくらいですか？

選択肢:

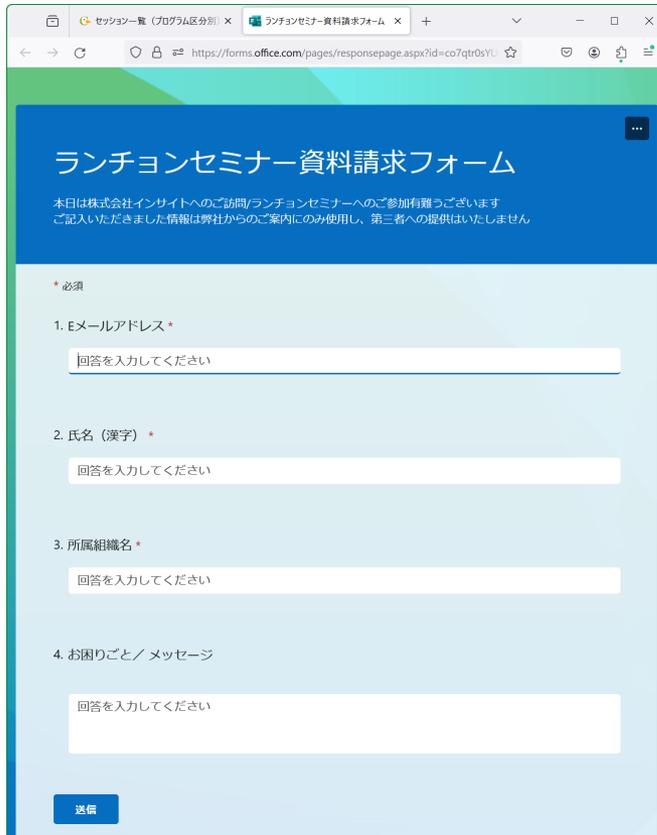
A. $\sigma_{\max} = \frac{300^4}{64 \times 20000}$

B. $\sigma_{\max} = \frac{300^2 \times 150^2}{64 \times 20000}$

C. $\sigma_{\max} = \frac{300^2 \times 150}{64 \times 20000}$

D. $\sigma_{\max} = \frac{300^4 \times 150}{64 \times 20000}$

資料請求(1)



The screenshot shows a web browser window with the URL <https://forms.office.com/pages/responsepage.aspx?id=co7qtr0&YL>. The page title is 'ランチョンセミナー資料請求フォーム' (Lunch Seminar Information Request Form). Below the title, there is a disclaimer: '本日は株式会社インサイトへのご訪問/ランチョンセミナーへのご参加有難うございます。ご記入いただきました情報は弊社からのご案内にのみ使用し、第三者への提供はいたしません。' (Thank you for your visit to InSight Co., Ltd./for participating in the Lunch Seminar today. The information you entered will only be used for our guidance, and we will not provide it to third parties.)

The form contains the following fields:

- 1. Eメールアドレス* (Email Address*)
回答を入力してください (Please enter your answer)
- 2. 氏名(漢字)* (Name (Kanji)*)
回答を入力してください (Please enter your answer)
- 3. 所属組織名* (Affiliation*)
回答を入力してください (Please enter your answer)
- 4. お困りごと/メッセージ (Inquiry/Message)
回答を入力してください (Please enter your answer)

A blue '送信' (Send) button is located at the bottom left of the form.

ランチョンセミナー資料請求フォーム



2. 生成AI チャット経験の紹介 (プログラミングや計算力学)(1)

- 質問1(Edge のBing Copilot)

ADVENTUREのmshフォーマットの有限要素四面体二次要素メッシュをVTKレガシーフォーマットに変換するpython3コードを書け

- 回答1

- Pythonのmeshioライブラリを使用して、Adventure mshフォーマットの有限要素四面体二次要素メッシュをVTKレガシーフォーマットに変換することができます。以下にそのコードを示します：

```
import meshio
# Adventure mshファイルの読み込み
mesh = meshio.read('input.msh')
# VTKフォーマットに変換して保存
meshio.write('output.vtk', mesh)
```

- このコードは、input.mshという名前のAdventure mshファイルを読み込み、同じメッシュデータをoutput.vtkという名前のVTKファイルに書き出します。
- ただし、このコードはmeshioライブラリがインストールされていることを前提としています。もしまだインストールされていない場合は、以下のコマンドでインストールできます：
- pip install meshio
- <後略>

2. 生成AI チャット経験の紹介 (プログラミングや計算力学)(2)

- 質問2 (Edge のBing Copilot)
- ADVENTURE mshフォーマットを誤解されているように思います。ADVENTURE mshのフォーマットは以下の通りです。「以下に四面体メッシュデータのフォーマットを示します。
- メッシュのコネクティビティについては 図 6.3-1 を参照してください。
- 節点番号, 要素番号は 0 から始まります。
- 拡張子は msh です。
- 170776 fl 全要素数
- 19900 19890 22150 22160 fl 1 件目の要素を構成する節点並び
- ~ 省略 ~
- 38274 38139 38266 38607 fl 170776 件目の要素を構成する節点並び
- 38608 fl 全節点数
- -31.223900 -3.384220 -5.000000 fl 1 件目の節点の座標
- ~ 省略 ~
- 31.223900 3.384220 5.000000 fl 38608 件目の節点の座標
- 2 fl ボリューム (領域) 数
- 2567 fl 第 1 ボリュームの要素数
- 0 fl 第 1 ボリュームの 1 件目の要素番号
- 1
- ~ 省略 ~
- 2566 fl 第 1 ボリュームの 2567 件目の要素番号
- ~ 省略 ~
- 注) 上記は四面体一次要素の場合, 二次要素の場合には要素のコネクティビティが 10 個となります。」
- <後略>

2. 生成AI チャット経験の紹介 (プログラミングや計算力学)(3)

- 回答2 (Edge のBing Copilot)
- 申し訳ありません、ADVENTURE mshフォーマットの詳細について誤解がありました。ご指摘いただきありがとうございます。このフォーマットについての新たな理解を元に、Pythonでの変換コードを改めて提供します。
- まず、ADVENTURE mshフォーマットのデータを読み込むための関数を作成します。その後、読み込んだデータをVTKフォーマットに変換して出力します。
- `import numpy as np`
- `import vtk`

```
def read_adventure_msh(filename):
    with open(filename, 'r') as f:
        lines = f.readlines()
        # 全要素数と全節点数を取得
    <略>
    # 要素を構成する節点並びを取得
    <略>
    # 節点の座標を取得
    <略>
    return np.array(nodes), np.array(elements)
```

```
def write_vtk(nodes, elements, filename):
    # VTKのUnstructuredGridオブジェクトを作成
    grid = vtk.vtkUnstructuredGrid()
    # 節点の座標を設定
```

<略>

```
# 要素を設定
```

<略>

```
# VTKファイルに書き出し
```

<略>

```
# ADVENTURE mshファイルを読み込み
```

```
nodes, elements =
```

```
read_adventure_msh('input.msh')
```

```
# VTKフォーマットに変換して保存
```

```
write_vtk(nodes, elements, 'output.vtk')
```

- このコードは、input.mshという名前のADVENTURE mshファイルを読み込み、同じメッシュデータをoutput.vtkという名前のVTKファイルに書き出します。

<後略>

2. 生成AI チャット経験の紹介 (プログラミングや計算力学)(4)

- 質問3 (Edge のBing Copilot) 0.0 0.0 1.0 fl 4 件目の節点の座標
- 四面体二次要素をADVENTURE mshのフォーマットで、極単純なメッシュデータを作って提示せよ 0.5 0.0 0.0 fl 5 件目の節点の座標
0.0 0.5 0.0 fl 6 件目の節点の座標
0.0 0.0 0.5 fl 7 件目の節点の座標
0.5 0.5 0.0 fl 8 件目の節点の座標
0.5 0.0 0.5 fl 9 件目の節点の座標
0.0 0.5 0.5 fl 10 件目の節点の座標
- 回答3 1 fl 全要素数 1 fl ボリューム (領域) 数
0 1 2 3 4 5 6 9 7 8 fl 1 件目の要素を構成する節点並び 1 fl 第 1 ボリュームの要素数
10 fl 全節点数 0 fl 第 1 ボリュームの 1 件目の要素番号
0.0 0.0 0.0 fl 1 件目の節点の座標
1.0 0.0 0.0 fl 2 件目の節点の座標
0.0 1.0 0.0 fl 3 件目の節点の座標

2. 生成AI チャット経験の紹介 (プログラミングや計算力学)(5)

- 質問4 (Edge のBing Copilot)
- ミーゼスの降伏条件を、誰が発案したかは別として、主応力で表現した、単位体積当たりの弾性歪エネルギーの内、剪断歪エネルギーが、材料固有の一定値に達した時に降伏するという考え方である事を、明解に解説せよ。
- 回答4
- ミーゼスの降伏条件は、材料の降伏理論の一つで、1913年にMisesが提案したものです³。この理論は、材料内の最大せん断ひずみエネルギーの値が、限界値に達すると降伏するという考え方を基にしています⁵。
- 具体的には、材料に働く応力が複雑で、多方向から荷重が加わるような状況でも、その応力状態を単位体積当たりの弾性歪エネルギーとして表現します²。そして、その弾性歪エネルギーのうち、剪断歪エネルギーが材料固有の一定値（降伏応力）に達した時に、材料が降伏すると考えます
- この降伏条件を数式で表すと、次のようになります⁵:

$$\sqrt{\frac{1}{2}[(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2]} = \sigma_{\text{Mises}} = \sigma_Y$$

- ここで、 σ_1 、 σ_2 、 σ_3 は主応力を、 σ_{Mises} はミーゼス応力（あるいは相当応力）、 σ_Y は降伏応力を表します⁵。
- この式は、3つの主応力による剪断歪エネルギーが降伏応力に等しくなる時、材料が降伏することを示しています⁵。つまり、この条件が満たされると、材料は弾性変形から塑性変形へと移行すると考えられます⁵。
- 以上が、ミーゼスの降伏条件についての基本的な説明です。この理論は、複雑な応力状態に対する材料の挙動を理解するための重要なツールとなっています²。ただし、実際の材料の挙動は、材料の種類や状態により異なるため、具体的な設計や解析には、適切な材料データや試験結果に基づく考慮が必要です¹。
- <後略>

2. 生成AI チャット経験の紹介 (プログラミングや計算力学)(6)

- 質問5 (Edge のBing Copilot)
- 以下のコードは、粒子が床で跳ね返る事は考慮するが、球同士の衝突を考慮していない。球同士の衝突と壁との衝突も考慮するように改造せよ。尚四方を囲む壁の位置は、明示的に与える事。粒子の初期配置は四方の壁の範囲内とする事。

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import FuncAnimation

class Particle:
    def __init__(self, mass, diameter, position,
velocity):
        self.mass = mass
        self.diameter = diameter
        self.position = np.array(position,
dtype=float)
        self.velocity = np.array(velocity,
dtype=float)
        self.force = np.zeros(3)
```

```
self.history = []

def compute_force(self, other):
    g = 9.8
    self.force = np.array([0, 0, -self.mass * g])

    for o in other:
        if o is not self:
            r = np.linalg.norm(self.position -
o.position)
            if r < (self.diameter + o.diameter) / 2:
                k, d, f = 1.0, 0.1, 0.1
                self.force += -k * (self.position -
o.position) - ¥
                self.velocity = d * (self.velocity - o.velocity) - f *
```

2. 生成AI チャット経験の紹介 (プログラミングや計算力学)(7)

- 質問5続き (Edge の Bing Copilot)

```
def update(self, dt):
    self.velocity += self.force / self.mass * dt
    self.position += self.velocity * dt
    # 床で跳ね返る
    if self.position[2] < 0:
        self.position[2] = 0
        # 反発係数を設定
        restitution_coefficient = 0.8
        self.velocity[2] = -restitution_coefficient *
self.velocity[2]
    self.history.append(self.position.copy().tolist())

def simulate(particles, dt, steps):
    for _ in range(steps):
        for p in particles:
            p.compute_force(particles)
        for p in particles:
            p.update(dt)

def animate(i):
    ax.cla()
```

```
for p in particles:
    ax.scatter(*p.history[i])

ax.set_xlim((-10., 10.))
ax.set_ylim((-10, 10))
ax.set_zlim((0, 10))
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.view_init(elev=10., azimuth=i*2)

# 粒子を作成
particles = [Particle(np.random.uniform(0.5, 1.5),
np.random.uniform(
    0.05, 0.15), np.random.uniform(-5, 5, 3), [0, 0, 0]) for i in
range(20)]
# シミュレーションを実行
simulate(particles, 0.02, 1000)
# 結果をプロット
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ani = FuncAnimation(fig, animate, frames=len(
    particles[0].history), interval=20)
plt.show()
```

2. 生成AI チャット経験の紹介 (プログラミングや計算力学)(8)

- 回答5

```
def update(self, dt):
    self.velocity += self.force / self.mass * dt
    self.position += self.velocity * dt
    # 床で跳ね返る
    if self.position[2] < box[0][2]:
        self.position[2] = box[0][2]
        # 反発係数を設定
        restitution_coefficient = 0.8
        self.velocity[2] = -restitution_coefficient *
self.velocity[2]
    # 壁で跳ね返る
    for i in range(3):
        if self.position[i] < box[0][i] or self.position[i] >
box[1][i]:
            self.velocity[i] = -restitution_coefficient *
self.velocity[i]
    self.history.append(self.position.copy().tolist())
```

3. Github copilotチャット 経験の紹介(1)

1. コメントを文脈から自動生成して呉れる

- 以下の4行が存在する。
 - `y_train_pred_label = np.argmax(y_train_pred, axis=1)`
 - `y_test_pred_label = np.argmax(y_test_pred, axis=1)`
 - `t_train_label = np.argmax(t_train, axis=1)`
 - `t_test_label = np.argmax(t_test, axis=1)`
- 先頭行の上に空行を作り、#oをタイプしただけで、意図を読み取り、
 - # one-hot表現からラベルに変換を補って呉れた。
- #だけだと、
 - # 予測結果をラベルに変換と補って呉れるのでTABキーで承認する。

3. Github copilotチャット 経験の紹介(2)

2. ログプリント文を文脈から自動生成して呉れる

以下の4行が存在する。

```
# 訓練正答の場合の画像と予測結果を表示
```

```
if train_correct_indices:
```

```
    draw_predit_and_image(  
        train_correct_indices, y_train_pred_label, x_train, '訓練正答')
```

```
最終行の後に空行を作り、else:迄入力すると、  
    my_learner.logger.info('train_correct_indices is empty.')
```

```
を提示して呉れる。
```

```
2行(ifブロック開始行)の行末にスペースを2個タイプすると、コメント  
を打ちたい意図を察知して呉れて
```

```
# 空でない場合
```

```
を補って呉れる。
```

3. Github copilotチャット 経験の紹介(3)

3. jsonファイルの読み込み

3.1 成功例

問1:

json ファイルの中身は以下の通りである。それを考慮して、全フィールドを適切な変数に代入せよ

```
{  
  "version": "1.21",  
  "distance": 4.0,  
  "yield_stress": 225,  
  "tensile_strength": 265,  
  "poisson_ratio": 0.3,  
  "youngs_modulus": 205800,  
  "hardening_coeff": 20580,  
  "adv_base_dir": "C:/tools/AdvOnWin0_43b",  
  "out_dir_base":  
  "../examples/beam_10x50x200_2",  
  "my_angle": 90,  
  "initial_n_steps": 20,  
  "gm3d_name": "beam_10x50x200_2.gm3d",  
  "thickness": 10.0,  
  "width": 50.0,
```

```
"length": 200.0,  
  "center_z_min": 100.0,  
  "center_z_max": 150.0,  
  "radius": 15.0,  
  "n_parametric_cases": 1,  
  "my_case_id": 101  
}
```

3.2 回答1

要望した全てのフィールドに対応する変数を作り、JSONオブジェクトから代入をして呉れた。非常に手間が省けた。

問2(失敗例)

data.jsonの中身を質問の中に貼り付けない時は、data.jsonがソースファイルと同じフォルダに在っても、それを読みに行く程親切では無い。

3. Github copilotチャット 経験の紹介(4)

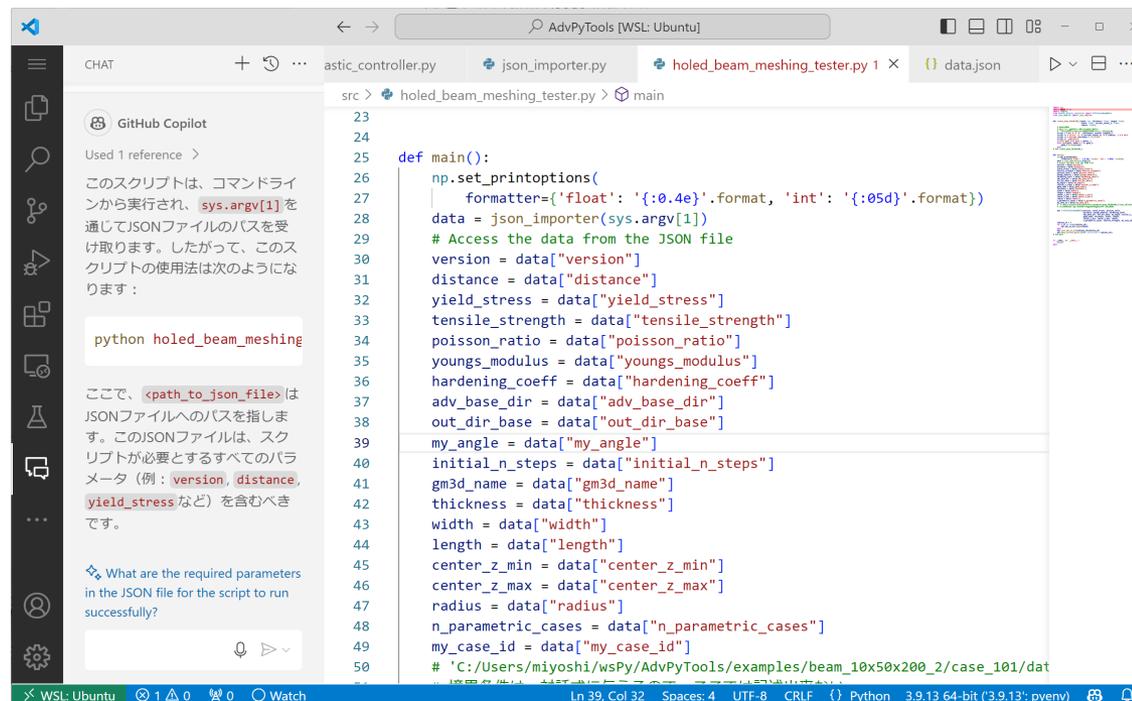
4. 実行時引数のUsageの作成

4.1 問1(失敗)

argv[1]の為のUsageを作成せよ(左のチャット用ブロック)

4.2 回答1

コードに直接追記する提案をして欲しかった。



The screenshot shows a code editor with a Python script and a GitHub Copilot chat window. The chat window contains the following text:

CHAT

Used 1 reference >

このスクリプトは、コマンドラインから実行され、`sys.argv[1]`を通じてJSONファイルのパスを受け取ります。したがって、このスクリプトの使用法は次のようになります:

```
python holed_beam_meshing
```

ここで、`<path_to_json_file>`はJSONファイルへのパスを指します。このJSONファイルは、スクリプトが必要とするすべてのパラメータ(例: `version`, `distance`, `yield_stress`など)を含むべきです。

What are the required parameters in the JSON file for the script to run successfully?

The code editor shows the following Python code:

```
def main():
    np.set_printoptions(
        formatter={'float': '{:0.4e}'.format, 'int': '{:05d}'.format})
    data = json_importer(sys.argv[1])
    # Access the data from the JSON file
    version = data["version"]
    distance = data["distance"]
    yield_stress = data["yield_stress"]
    tensile_strength = data["tensile_strength"]
    poisson_ratio = data["poisson_ratio"]
    youngs_modulus = data["youngs_modulus"]
    hardening_coeff = data["hardening_coeff"]
    adv_base_dir = data["adv_base_dir"]
    out_dir_base = data["out_dir_base"]
    my_angle = data["my_angle"]
    initial_n_steps = data["initial_n_steps"]
    gm3d_name = data["gm3d_name"]
    thickness = data["thickness"]
    width = data["width"]
    length = data["length"]
    center_z_min = data["center_z_min"]
    center_z_max = data["center_z_max"]
    radius = data["radius"]
    n_parametric_cases = data["n_parametric_cases"]
    my_case_id = data["my_case_id"]
    # 'C:/Users/miyoshi/wsPy/AdvPyTools/examples/beam_10x50x200_2/case_101/dat
```

3. Github copilotチャット 経験の紹介(5)

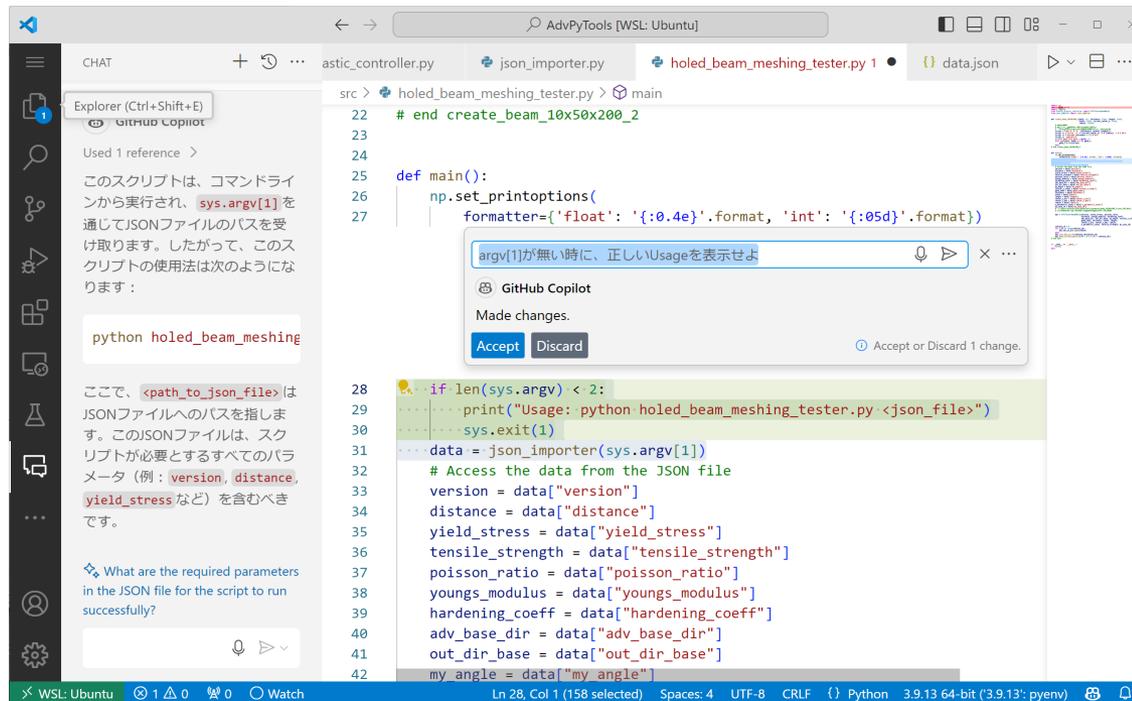
4. 実行時引数のUsageの作成

4.3 問2

argv[1]が無い時に、正しいUsageを表示せよ

4.3 回答2

満足。



The screenshot shows a VS Code editor window with a Python script named `holed_beam_meshing_tester.py`. The script includes a `main()` function that uses `json_importer.py` to process JSON data. A GitHub Copilot chat window is open on the left, displaying a conversation about handling command-line arguments. The chat window shows a user prompt: "argv[1]が無い時に、正しいUsageを表示せよ" (When argv[1] is missing, display the correct usage). The Copilot response includes a code snippet for a usage message: `print("Usage: python holed_beam_meshing_tester.py <json_file>")`. The chat window also shows a confirmation that the changes were made and an option to accept or discard the change.

```

src > holed_beam_meshing_tester.py > main
22 # end create_beam_10x50x200_2
23
24
25 def main():
26     np.set_printoptions(
27         formatter={ 'float': '{:0.4e}'.format, 'int': '{:05d}'.format})
28
29
30
31     data = json_importer(sys.argv[1])
32     # Access the data from the JSON file
33     version = data["version"]
34     distance = data["distance"]
35     yield_stress = data["yield_stress"]
36     tensile_strength = data["tensile_strength"]
37     poisson_ratio = data["poisson_ratio"]
38     youngs_modulus = data["youngs_modulus"]
39     hardening_coeff = data["hardening_coeff"]
40     adv_base_dir = data["adv_base_dir"]
41     out_dir_base = data["out_dir_base"]
42     my_angle = data["my angle"]

```

CHAT

Used 1 reference >

このスクリプトは、コマンドラインから実行され、`sys.argv[1]`を通じてJSONファイルのパスを受け取ります。したがって、このスクリプトの使用法は次のようになります:

```
python holed_beam_meshing
```

ここで、`<path_to_json_file>`はJSONファイルへのパスを指します。このJSONファイルは、スクリプトが必要とするすべてのパラメータ (例: `version`, `distance`, `yield_stress` など) を含むべきです。

What are the required parameters in the JSON file for the script to run successfully?

argv[1]が無い時に、正しいUsageを表示せよ

GitHub Copilot

Made changes.

Accept Discard

Accept or Discard 1 change.

3. Github copilotチャット 経験の紹介(6)

4. 実行時引数のUsageの作成

4.3 問2

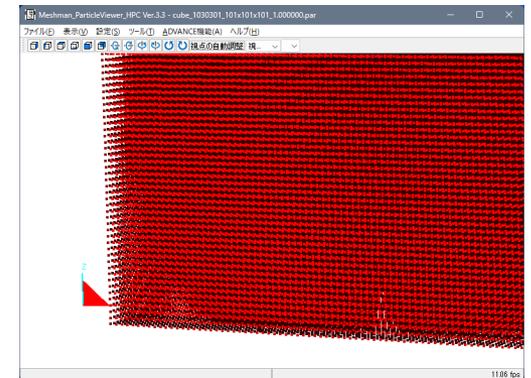
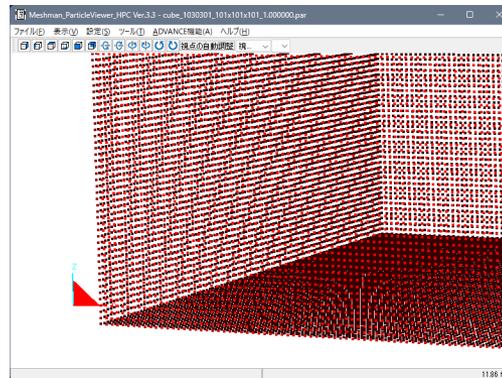
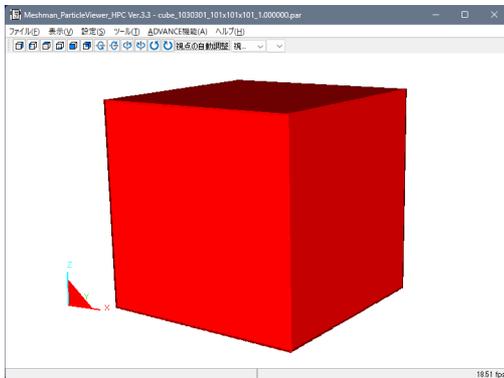
argv[1]が無い時に、正しいUsageを表示せよ

4.3 回答2

満足。

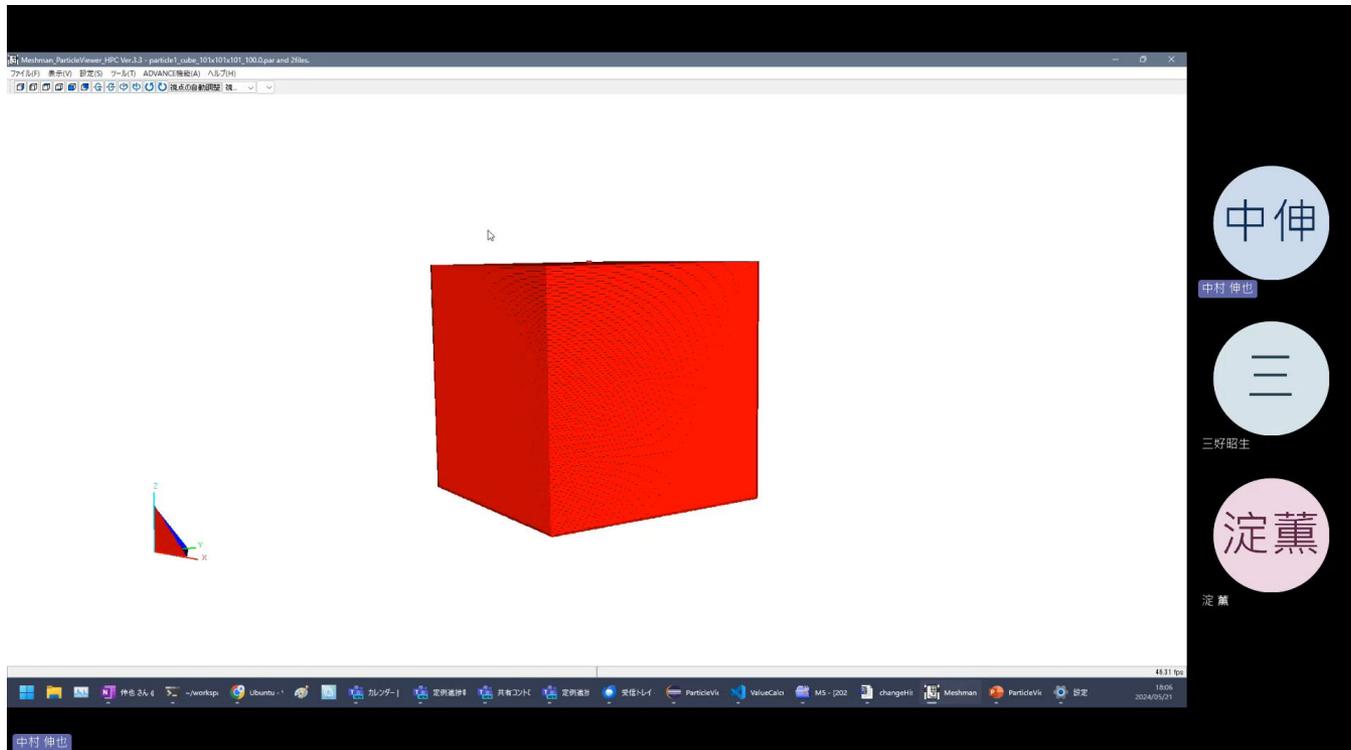
4. Meshman_ParticleViewer_HPC の最新版紹介(1)

- 無限zoom
 - どれだけ拡大しても、手前の面でモデルが切られる事が無い。
 - 又、拡大時にはマウス操作感度が調節され低くなる。
- 粒子データでも、陰影で立体的に描画する
- 粒子データの表面を抽出し、描画を軽くする
 - 全ての粒子を表示する事も可能



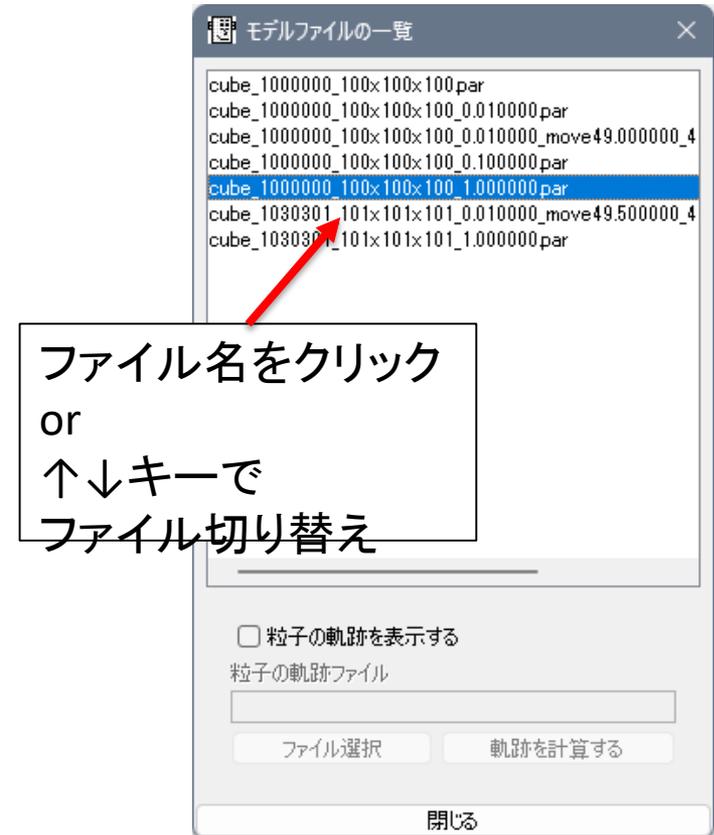
4. Meshman_ParticleViewer_HPC の最新版紹介(2)

- 無限zoom



4. Meshman_ParticleViewer_HPC の最新版紹介(3)

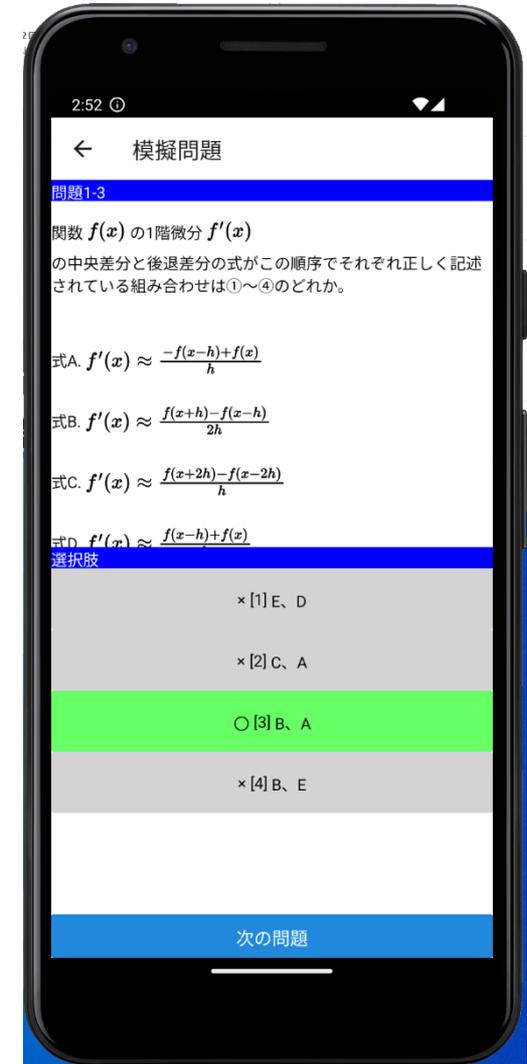
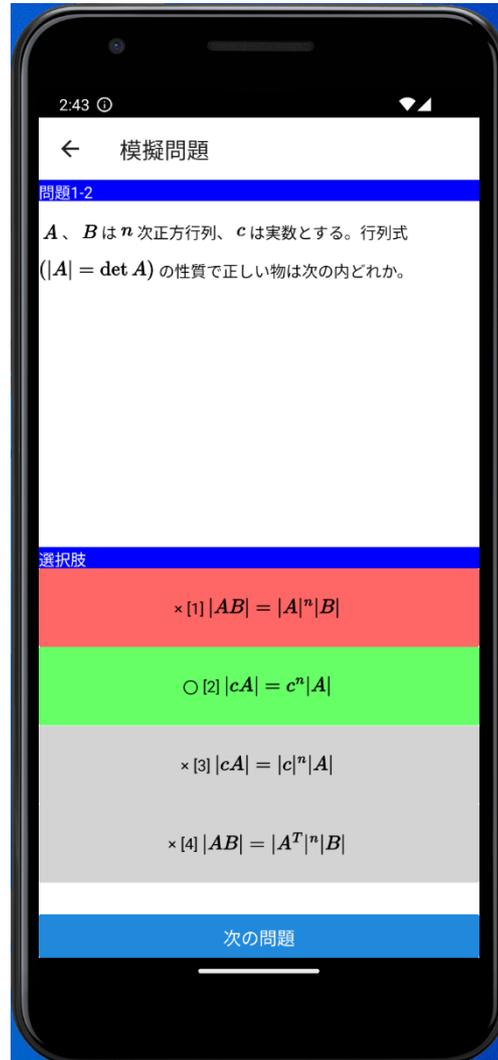
- バグ修正(2024/05)
 - 問題点：
時系列データなどを可視化する際に、ファイルリストのダイアログを使用する。そのダイアログ内で表示するファイルの切り替えを行うと、フォーカスがメインウィンドウに移動してしまっていた。
 - 修正：
フォーカス移動の処理を修正し、表示するファイルを切り替えた際に、ファイルリストのダイアログからフォーカスが移動しないようにした。これにより、方向キーの操作で連続的にファイルを切り替えられるようになった。



ファイルリストのダイアログ

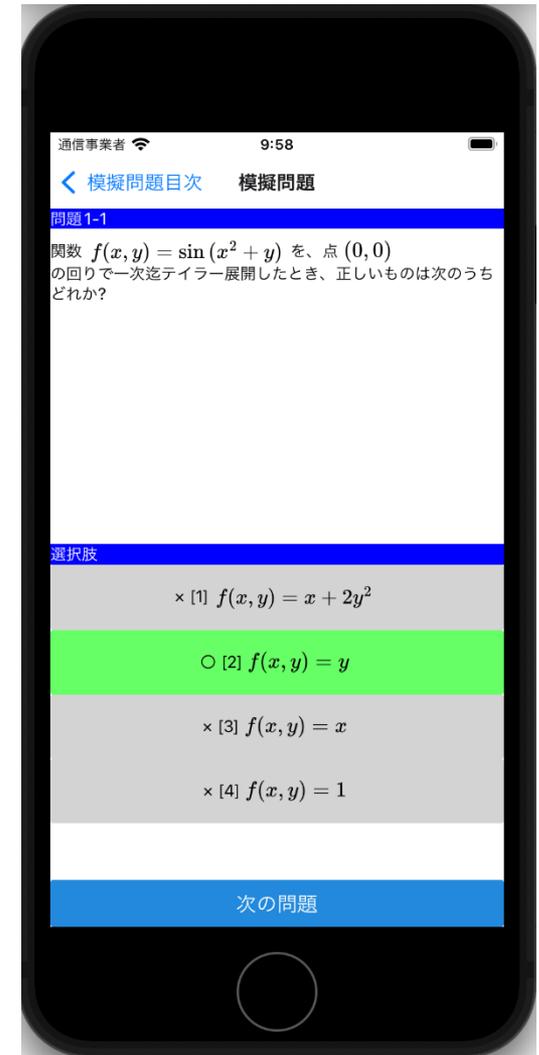
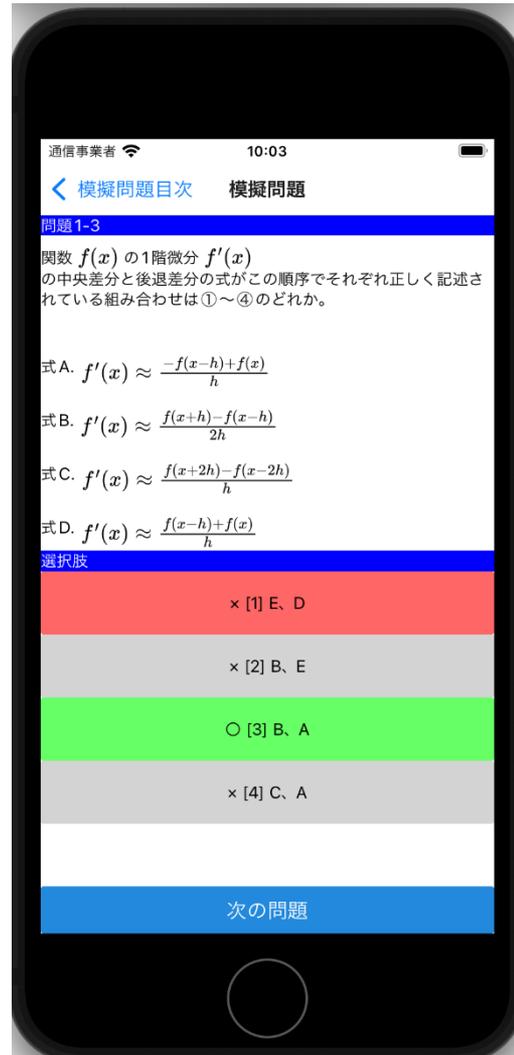
5. 計算力学2級スマホアプリの 更新版紹介(1)

- Android版



5. 計算力学2級スマホアプリの 更新版紹介(2)

- iPhone版



6. Meshman_ParticlePackingと OSSのDEMアプリ(1)

- Meshman_ParticlePackingが粒子初期位置を生成
- 後工程でDEM(離散要素法)解析
- OSSのYade
 - Python I/Fを利用してYadeを操作
 - 熱伝導解析に成功

6. Meshman_ParticlePackingと OSSのDEMアプリ(2)

Chat-Gpt4o

問:OSS のDEMアプリYADEでHooper Binから粉体を落とす解析を行いたい。
Python Interfaceで典型例を記述せよ

回答:

```
from yade import pack, plot
# Create simulation scene
O.bodies.append(geom.facetBox((0,0,0), (1,1,1), wallMask=31)) # Boundary
box

# Create Hooper Bin using facets
hooperBin = [
    facet([(-0.5, -0.5, 0), (0.5, -0.5, 0), (0, 0, 0.5)]),
    facet([(0.5, -0.5, 0), (0.5, 0.5, 0), (0, 0, 0.5)]),
    facet([(0.5, 0.5, 0), (-0.5, 0.5, 0), (0, 0, 0.5)]),
    facet([(-0.5, 0.5, 0), (-0.5, -0.5, 0), (0, 0, 0.5)])
]
O.bodies.append(hooperBin)

# Define material for particles
sphereMat = FrictMat(young=1e7, poisson=0.3, frictionAngle=radians(30),
density=2600)

# Create particles (spheres)
sp = pack.SpherePack()
sp.makeCloud((-0.4,-0.4,0.5), (0.4,0.4,0.9), rMean=0.05, rRelFuzz=0.3)
sp.toSimulation(material=sphereMat)

# Simulation engines
```

```
O.engines = [
    ForceResetter(),
    InsertionSortCollider([Bo1_Sphere_Aabb(), Bo1_Facet_Aabb()]),
    InteractionLoop(
        [Ig2_Sphere_Sphere_ScGeom(), Ig2_Facet_Sphere_ScGeom()],
        [Ip2_FrictMat_FrictMat_FrictPhys()],
        [Law2_ScGeom_FrictPhys_CundallStrack()]
    ),
    NewtonIntegrator(gravity=(0,0,-9.81), damping=0.4),
    PyRunner(command='checkUnbalanced()', realPeriod=2, label='checker')
]

# Function to check unbalanced force
def checkUnbalanced():
    if O.iter < 5000: return
    if utils.unbalancedForce() < 0.05:
        O.pause()
        plot.saveDataTxt('hooperBin.dat.bz2')

# Start simulation
O.dt = utils.PWaveTimeStep()
O.saveTmp()
O.run()
# Visualization
from yade import qt
qt.Controller()
qt.View()
```

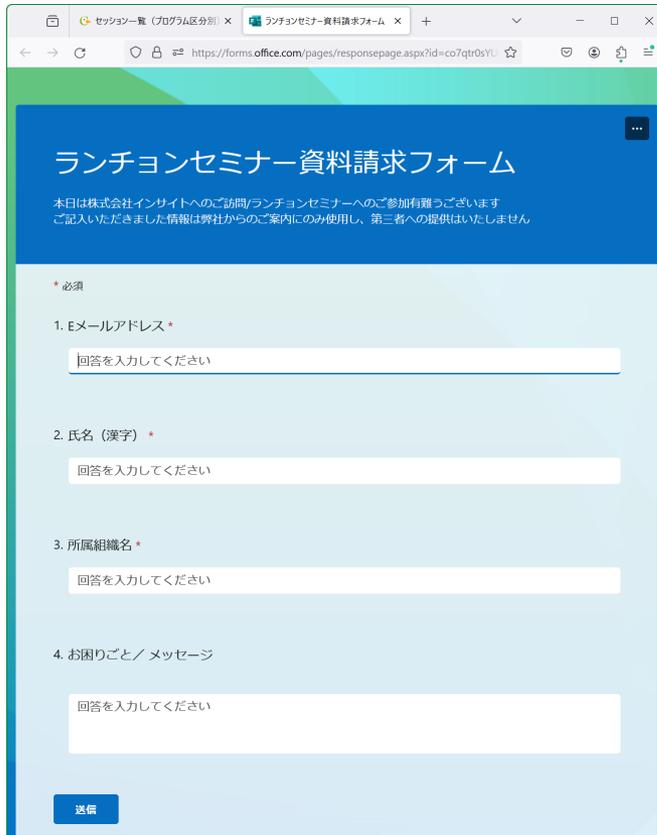
7. データサイエンス勉強会の 更新版紹介(1)

- 深層学習シリーズ
 - 深層学習1
 - 深層学習2
 - レベル6:学習の枠組み
 - レベル7:2層NNの学習の実装
 - CAEへの適用事例
 - 深層学習3

7. データサイエンス勉強会の 更新版紹介(2)

- 深層学習2
 - 参考実装のソースのバグフィックス
 - 数値微分を利用した微分
 - 分類が正常に行われるようになった
 - 計算が遅い為デバッグに著しい工数が掛かっていた。
 - Bing Copilotを利用してバグフィックスを行った

資料請求(1)



The screenshot shows a web browser window with the following content:

- Browser tabs: セッション一覧 (プログラム区分別) ×, ランチョンセミナー資料請求フォーム ×
- Address bar: <https://forms.office.com/pages/responsepage.aspx?id=co7qtr0&YL>
- Form Title: ランチョンセミナー資料請求フォーム
- Introductory text: 本日は株式会社インサイトへのご訪問/ランチョンセミナーへのご参加有難うございます。ご記入いただきました情報は弊社からのご案内にのみ使用し、第三者への提供はいたしません。
- Form Fields:
 - * 必須
 - 1. Eメールアドレス *
回答を入力してください
 - 2. 氏名 (漢字) *
回答を入力してください
 - 3. 所属組織名 *
回答を入力してください
 - 4. お困りごと / メッセージ
回答を入力してください
- Submit Button: 送信

ランチョンセミナー資料請求フォーム

